Multicore Performance of Block Algebraic Iterative Reconstruction Methods

Sørensen, Hans Henrik B.; Hansen, Per Christian

Published in:
S I A M Journal on Scientific Computing

Link to article, DOI:
10.1137/130920642

Publication date:
2014

Document Version
Peer reviewed version

MULTI-CORE PERFORMANCE OF BLOCK ALGEBRAIC ITERATIVE RECONSTRUCTION METHODS

HANS HENRIK B. SØRENSEN* AND PER CHRISTIAN HANSEN*

Abstract. Algebraic iterative methods are routinely used for solving the ill-posed sparse linear systems arising in tomographic image reconstruction. Here we consider the Algebraic Reconstruction Techniques (ART) and the Simultaneous Iterative Reconstruction Techniques (SIRT), both of which rely on semi-convergence. Block versions of these methods, based on a partitioning of the linear system, are able to combine the fast semi-convergence of ART with the better multi-core properties of SIRT. These block methods separate into two classes: those that, in each iteration, access the blocks in a sequential manner, and those that compute a result for each block in parallel and then combine these results before the next iteration. The goal of this work is to demonstrate which block methods are best suited for implementation on modern multi-core computers. To compare the performance of the different block methods we use a fixed relaxation parameter in each method, namely, the one that leads to the fastest semi-convergence. Computational results show that for multi-core computers, the sequential approach is preferable.

Key words. algebraic iterative reconstruction, ART, SIRT, block methods, relaxation parameter, semi-convergence, tomographic imaging

AMS subject classifications. 65F10 (Iterative methods for linear systems), 65R32 (Inverse problems)

1. Introduction. Discretizations of tomographic imaging problems often lead to large sparse systems of linear equations with noisy data:

$$A\,x \simeq b, \qquad b = \bar{b} + e, \qquad A \in \mathbb{R}^{m\times n}. \tag{1.1}$$

Here $\bar{b} = A\,\bar{x}$ denotes the exact data, $\bar{x}$ is the exact solution, and e is the perturbation consisting of additive noise. The sparse matrix A models the forward problem; there are no restrictions on its dimensions, and both over- and under-determined systems arise in applications — depending on the amount of data in a given experiment. For examples of such system, see [3], [4], [29], [40].

Iterative algorithms are ideal for solving the large-scale problem (1.1) [30] and several classes of methods have emerged [19]. They have in common that they produce regularized solutions to problems with noisy data, i.e., solutions that approximate the exact and unknown solution $\bar{x}$ without being too sensitive to the perturbation e. This work focuses on two specific classes of algebraic iterative methods, both of which are often used in tomographic imaging: the Algebraic Reconstruction Technique (ART) and the Simultaneous Iterative Reconstruction Technique (SIRT), and in particular block extensions of these two methods. In our approach, we assume that the matrix A is generated prior to the reconstruction procedure and provided as input data, in contrast to matrix-free approaches.

The number of computing cores on contemporary computer platforms increases rapidly, which spawns an increased need for parallelism. The importance of utilizing the memory hierarchy for data locality makes it more difficult to choose the most appropriate algebraic reconstruction algorithm for a given setup. This work rigorously investigates the multi-core computational performance of block algebraic iterative

*Department of Applied Mathematics and Computer Science, Technical University of Denmark, DK-2800 Lyngby, Denmark ({hhbs,pcha}@dtu.dk). This project is supported by grants no. 274-07-0065 and 09-070032 from the Danish Research Council for Technology and Production Sciences.

reconstruction methods using the OpenMP framework; see [34] for an MPI-based application on a heterogeneous computing cluster.

The goal of our work is to study the performance of the block methods on state-of-the-art hardware, where block operations are preferred. To do so we describe two different ways to obtain block methods, and we show how they relate to the standard ART and SIRT methods and how they encompass various block methods that have already been described in the literature. We also present a thorough comparison of these methods on realistic model problems, with emphasis on computational performance and computing speed. Some early works on block methods are [2], [7], [14], and [48]. A specific study of the block approach in the Diagonally Relaxed Orthogonal Projection (DROP) method can be found in [12].

Our paper gives a framework for describing block algebraic iterative methods, and in this respect it is similar in spirit to [32] and [44]. Our emphasis, however, is on implementation aspects, and our main contribution is a careful comparison of the methods in which we pay attention to core factors that influence the convergence speed — such as the row ordering, the block size, structural orthogonality of the rows, and the choice of the relaxation (step-length) parameter. We also study the role played by blocks consisting of structurally orthogonal rows. All our algorithms are implemented in C and carefully tuned for optimal cache usage and for multi-threading using OpenMP. We present numerical experiments on two different multi-core platforms for a variety of block partitionings.

We are solely concerned with the semi-convergence of the methods when applied to noisy data (asymptotic convergence is a different issue). There are several techniques for adjusting the relaxation parameter in each iteration (see, e.g., [28]); but including a thorough study of these non-stationary methods would only distract from our main focus – the handling of the blocks. Instead we use a carefully chosen fixed relaxation parameter for each method. For the same reason we do not consider stopping criteria.

The paper is organized as follows. We start with a summary of the basic algebraic iterative reconstruction methods in Section 2, where we also discuss the choice of the relaxation parameter. In Section 3 we present the framework for the block methods and introduce important special cases, and we discuss the influence of structurally orthogonal rows. The main results are presented in Section 4 where we compare the performance of the block methods on two large-scale model problems. An appendix summarizes row ordering techniques and their influence on the convergence and, in particular, on the choice of the blocks.

2. Algebraic Iterative Reconstruction Methods. In this section we briefly summarize the basic algebraic iterative methods and set the notation and ideas for the following chapters. Throughout, a_i^T denotes the ith row of A, λ is a relaxation parameter, and $P_{\mathcal{C}}$ denotes the orthogonal projection on the convex set $\mathcal{C}$ — such as the positive orthant giving nonnegative solutions. We also discuss our strategy for computing the optimal fixed value of the relaxation parameter λ. We refer to [20], [29], [40] for more details and convergence analysis. The reader should take note that there is no consistent naming convention across the imaging literature for all the methods considered here; we take care to define which methods we consider, and give a summary of the methods, with references, in Table 3.1.

2.1. The Basic Methods. ART (Algebraic Reconstruction Technique) methods are inherently sequential, in that they treat the rows one at a time in a certain order (the difference between the methods lies in the choice of this order; see [24] for the original work). In the literature, "ART" may refer to slightly different methods

with and without projections; here we use it synonymously with the classical Kaczmarz method [33] augmented with a projection $P_{\mathcal{C}}$, where the kth iteration involves a sweep over the rows of A:

Algorithm: $x^k \leftarrow$ **ART-SWEEP** (λ, A, b, x^{k-1})

$$x^{k,0} = x^{k-1}$$
$$x^{k,i} = P_{\mathcal{C}}\left(x^{k,i-1} + \lambda \, \frac{b_i - a_i^T \, x^{k,i-1}}{\|a_i\|_2^2} \, a_i \right), \qquad i = 1, \ldots, m$$
$$x^k = x^{k,m}.$$

The asymptotic convergence of this method requires $0 < \lambda < 2$ and is governed by $\mathrm{cond}(A)$, the condition number of A; e.g., if the system is square and has full rank, $\lambda = 1$, and A has unit rows that are chosen randomly with equal probability then

$$\mathcal{E}\big(\|x^k - x^*\|_2^2\big) \leq \left(1 - \frac{1}{\mathrm{cond}(A)^2}\right)^k \|x^0 - x^*\|_2^2 \, , \tag{2.1}$$

where $x^* = A^{-1} \, b$ and $\mathcal{E}(\cdot)$ denotes the expected value [47]. In particular, if all rows are orthogonal then ART converges in one sweep (this follows from (2.1) when the rows are orthonormal $\Rightarrow \mathrm{cond}(A) = 1$). See [16] for a more general treatment of the convergence. ART methods are recognized for having fast initial convergence when used for image reconstruction problems if λ is chosen properly.

SIRT (Simultaneous Iterative Reconstruction Technique) methods involve all the rows of A simultaneously, and thus involve matrix-vector products. Similar to ART, the term "SIRT" refers to slightly different methods in the literature; this paper follows [49] and SIRT denotes methods[1] where the kth iteration takes the generic form

$$x^k = P_{\mathcal{C}}\big(\, x^{k-1} + \lambda \, T A^T M \, (b - A \, x^{k-1}) \, \big) \ . \tag{2.2}$$

Here T and M are positive definite matrices, and asymptotic convergence requires that that $0 < \lambda < 2/\|T^{-1/2} A M^{1/2}\|_2^2$. By selecting different T and M we obtain well-established SIRT methods. Landweber's method [36] corresponds to $T = I$ and $M = I$, and Cimmino's method[2] [6] is obtained by $T = I$ and $M = \mathrm{diag}(1/\|a_i\|_2^2)$ corresponding to normalizing the rows of A. The CAV (component averaging) method [9] corresponds to $T = I$ and $M = \mathrm{diag}(1/\sum_{j=1}^n \nu_j a_{ij}^2)$, where ν_j is the number of nonzeros in the jth column of A. The single-block DROP method [9, 12] is defined by the same M as for Cimmino's method and with $T = \mathrm{diag}(m/\nu_j)$. In 1972 Gilbert [21] defined an algorithm called "SIRT" which differs from our SIRT methods.

Note that if $\mathrm{rank}(A) = m$ and we choose $M = (AA^T)^{-1}$ then $A^T M$ is the pseudoinverse of A and (2.2) converges in one step if $P_{\mathcal{C}}$ is the identity. The different SIRT algorithms correspond to different diagonal approximations to $(AA^T)^{-1}$; e.g., we obtain Cimmino with $M = \mathrm{diag}(AA^T)^{-1}$.

The asymptotic convergence of SIRT is governed by $\mathrm{cond}(M^{1/2} A)$; it follows from Thm. 2.1.15 in [42] that if we choose λ slightly smaller than $2/\|M^{1/2} A\|_2^2$ then

$$\|x^k - x^*\|_2 \lesssim \left(\frac{\mathrm{cond}(M^{1/2} A)^2 - 1}{\mathrm{cond}(M^{1/2} A)^2 + 1}\right)^k \|x^0 - x^*\|_2 \, , \tag{2.3}$$

[1] These methods might also be called "Cimmino-type methods."

[2] A scaling factor $1/m$ or $2/m$ is sometimes included in M.

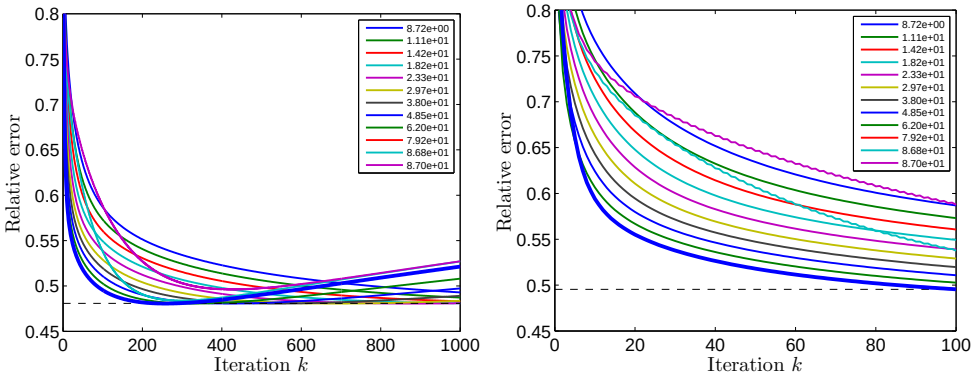


FIG. 2.1. *Typical errors histories* $\|\bar{x} - x^k\|_2/\|\bar{x}\|_2$ *for different choices of* λ. *The dashed line corresponds to the smallest relative error within* $k_{\max}$ *iterations. Left:* $k_{\max} = 1000$ *is large enough that semi-convergence is observed for a range of* λ*-values. Right:* $k_{\max} = 100$ *is not large enough that semi-convergence can take place, and the minimum error is attained for* $k = k_{\max}$. *In both cases, the thick line shows the error history corresponding to the optimal* λ.

where x^* is the solution to $\min_x \|M^{1/2}(A\,x - b)\|_2$. See [15] for a summary of asymptotic convergence properties and references to original papers.

For all ART and SIRT methods we use the starting vector $x^0 = 0$. There is no evidence that other choices lead to any significant improvement in the reconstructions or savings in computing time; see, e.g., [30], [50].

The basic computational task of an ART iteration is a sequence of consecutive inner products. Since the number of nonzero elements in a row a_i of A is small it is difficult to implement this method efficiently on modern computer platforms; it does not lend itself well to parallel implementation, neither can it take advantage of the block matrix operations that typically increase computational speed. In contrast, the basic computational step in the SIRT methods is composed of two matrix-vector operations, which lend themselves to efficient implementation on current hardware.

2.2. The Influence of the Relaxation Parameter. ART and SIRT exhibit *semi-convergence* [40] when applied to noisy data. During the first iterations the iterates approach the exact solution $\bar{x}$, and after this stage the iterates start to diverge from $\bar{x}$ and instead converge to a solution that is dominated by noise, cf. [16], [18], [27].

The relaxation parameter λ plays an important role during the semi-convergence phase. The choice of λ is critical; if it is too small then we have very slow convergence, and if it too large then the process exhibits neither convergence nor semi-convergence. For inconsistent problems a diminishing step-size strategy is needed to ensure asymptotic convergence, while a fixed λ still provides semi-convergence (see, e.g., [15], [18]).

Figure 2.1 illustrates the influence of λ; the different curves are the error histories $\|\bar{x} - x^k\|_2/\|\bar{x}\|_2$ for different choices of λ, and the dashed line is the smallest relative error obtained within $k_{\max}$ iterations. To emphasize the semi-convergence behavior we use a small 2D parallel-beam test problem with a 32×32 Shepp-Logan phantom, 36 projections of 32 pixels each, and 5% noise in the data (generated with AIR Tools [28]). Assume first that $k_{\max}$ is large enough that semi-convergence takes place for a range of λ-values, as shown in the left plot. The thick line shows the error history corresponding to the optimal λ, i.e., the one that reaches the smallest relative error

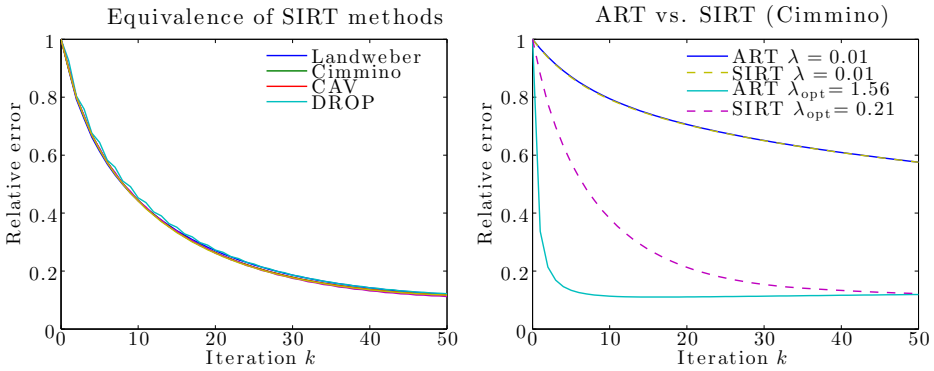


FIG. 2.2. *Error histories of the different basic methods (we used the small test problem described in §4.1). The left figure shows a comparison of different SIRT methods for optimal choices of the fixed relaxation parameter λ. The right figure shows a comparison of ART and SIRT for a very small fixed relaxation parameter of $\lambda = 0.01$ and for optimal choices of the fixed relaxation parameter.*

0.48 in the smallest number of iterations $k \approx 250$. Note that some of the error histories (for the largest values of λ) do not reach the minimum attainable error.

Now assume that $k_{\max}$ is so small that semi-convergence does not take place, i.e., all errors histories have a minimum at $k = k_{\max}$; then the optimal λ is the one that reaches the smallest relative error 0.495 at $k = k_{\max}$. Note that the relative error here is larger than for the case where $k_{\max}$ is large enough to provide semi-convergence.

Figure 2.2 adds further insight into the dependence of the algorithms on λ. The left part shows the error history for four SIRT methods, using the optimal relaxation parameter for each method. All methods have almost identical convergence histories, and therefore we need only consider the Cimmino method — denoted SIRT from now on.

The right part of Fig. 2.2 compares the convergence histories for ART and SIRT. For very small values of λ we observe that the convergence of both methods is practically identical; this is consistent with the conclusions in [48]. If we choose the optimal λ for each method, then ART converges considerably faster than SIRT, in the sense that ART achieves the minimum error in far fewer iterations than SIRT. We note the relation between, on the one hand, ART and SIRT and, on the other hand, the Jacobi and Gauss-Seidel iterative schemes, where a similar difference in convergence rate is observed.

2.3. Computing the Optimal Fixed Relaxation Parameter. Recall that we define the optimal value of λ for a given method as the one that reaches the smallest error in the smallest number of iterations; such a value only exists for problems that give rise to semi-convergence, i.e., problems with noisy data. Our approach to computing the optimal λ extends the training technique from [28].

The key idea in this technique is to find the parameter that, for noisy data, minimizes the relative error $\|\bar{x} - x^k\|_2 / \|\bar{x}\|_2$ in the smallest number of iterations, within a maximum number of iterations $k_{\max}$ (this obviously requires that we have access to a model problem for which the exact solution $\bar{x}$ is known). We note that if semi-convergence occurs within $k_{\max}$ iterations, then for the SIRT methods the minimum error is practically independent of λ [18]. While the same is sometimes observed for the ART methods, this is not guaranteed to always be the case.

Here we outline a training procedure to target a specific relative error η_{target}

within a small tolerance in a robust manner. The user must specify the matrix A, the noisy right-hand side b, the exact solution $\bar{x}$, and the maximum number of iterations $k_{\max}$. Our algorithm to compute the optimal relaxation parameter λ_{opt} then takes the form:

Algorithm: ORP (Optimal Relaxation Parameter)

1. Initialization:
   (a) Find the λ in the permissible range for convergence that gives the smallest relative error $\eta_{\min}$ within $k_{\max}$ iterations. Denote this λ by $\lambda_{\min}$.
   (b) Set the target relative error $\eta_{\text{target}} \geq \eta_{\min}$.
2. Find the smallest $\lambda < \lambda_{\min}$ that, within a small tolerance, reaches the target error η_{target} in at most $k_{\max}$ iterations. Denote this λ by $\underline{\lambda}$.
3. Find the largest $\lambda > \lambda_{\min}$ that, within a small tolerance, reaches the target error η_{target} in at most $k_{\max}$ iterations. Denote this λ by $\bar{\lambda}$.
4. Find the $\lambda \in [\underline{\lambda}, \bar{\lambda}]$ that, within a small tolerance, reaches the target error η_{target} in the smallest number of iterations, and set λ_{opt} equal to this λ.

To fairly compare different (block) algebraic methods on a given problem using an optimal fixed relaxation parameter λ_{opt}, we make sure that all the methods of interest are able to reach, within a small tolerance, the same $\eta_{\min}$. This way, no method will be considered unless it is able to reach the same minimum relative error as all the other methods.

3. Block Methods. In this section, we give an overview and a framework for block methods based on the above ART and SIRT methods. The basic idea of the block methods is to *partition* the matrix A and the right-hand side b of (1.1) into a total of p of blocks. We write

$$A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_p \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{pmatrix}, \qquad A_\ell \in \mathbb{R}^{m_\ell \times n}, \quad \ell = 1, \ldots, p, \tag{3.1}$$

where m_ℓ is the number of rows of the ℓth block A_ℓ. For more details about the block methods and their convergence, see, e.g., [7], [13], [17], [32], [41].

Following to the classification in [7] we distinguish between methods that access the blocks either sequentially or in parallel. The former methods, in each iteration, access the blocks in a sequential manner, while the latter methods compute a result for each blocks in parallel and then combine these results before the next iteration. In this context, one iteration of a block method is performed when all blocks have been processed — either in sequence or in parallel. The distinction between the two types of methods precisely reflects the structure of the basic ART and SIRT methods described in the previous section. Table 3.1 gives a convenient overview of the algorithms together with some original names and relevant references.

3.1. Block-Sequential Methods. Common for these methods is that they treat the blocks of A sequentially (similar to how ART treats the rows of A). How we specifically treat the blocks defines the block-sequential method.

Consider first the situation where we sequentially apply a SIRT method to each block. In the kth iteration, the input to the treatment of the first block is the previous iteration vector x^{k-1}, and it produces the intermediate vector $x^{k,1}$. This vector, then, is used as the input for the treatment of the second block, producing the vector

TABLE 3.1
Overview of the algorithms considered with references to relevant literature. "Outer" and "Inner," respectively, refer to the access of the blocks and how each block is treated. "Pinv" denotes the use of a pseudo-inverse A_ℓ^+.

Name	Outer	Inner	Reference
ART	Seq.	ART	ART [24], Kaczmarz [33]
BLOCK-IT	Seq.	SIRT	[8], Block Cimmino [1], Block Iteration [17]
SEQ-PINV	Seq.	Pinv	"Gauss-Seidel type" [14], "Generalized Kaczmarz" [13]
PART	Seq.	ART	PART [22]
SIRT	Par.	SIRT	Cimmino [6], CAV [9], DROP [12], Landweber [36]
SAP	Par.	ART	String-Averaging-Projections [11]
CARP	Par.	ART	CARP (Component-Averaged Row Projections) [23]
PAR-PINV	Par.	Pinv	"Jacobi type" [14]

$x^{k,2}$ — which, in turn, is used as input vector to the third block, an so on. Once all p blocks have been processed, the resulting vector $x^{k,p}$ is then labeled as the kth iteration vector x^k. We emphasize that the projection $P_{\mathcal{C}}$ introduced above can be incorporated in the computation of each $x^{k,\ell}$ for $\ell = 1, \ldots, p$.

The algorithm[3] thus takes the following form, where the matrices T (which is computed from A) and $M_\ell \in \mathbb{R}^{m_\ell \times m_\ell}$, $\ell = 1, \ldots, p$ define which SIRT method is used for each block:

Algorithm: BLOCK-IT
Initialization: choose an arbitrary $x^0 \in \mathbb{R}^n$
Iteration: for $k = 0, 1, 2, \ldots$

$$x^{k,0} = x^{k-1}$$
$$x^{k,\ell} = P_{\mathcal{C}}\big(x^{k,\ell-1} + \lambda\, T\, A_\ell^T\, M_\ell\, (b_\ell - A_\ell\, x^{k,\ell-1}) \big), \quad \ell = 1, 2, \ldots, p$$
$$x^k = x^{k-1,p}.$$

Using ART as the basis for handling each block is not relevant, because the overall algorithm then becomes the standard ART method. BLOCK-IT [1], [17] was considered in several papers, e.g., [7], [48]. A special instance of this algorithm is SART [2], which is an early example of a block-sequential method that uses a fixed partitioning (each block corresponds to data associated with a "projection" in parallel-beam tomography).

An alternative to the above SIRT-based method is due to Elfving [14], who proposed a block iterative method with $T = I$ and $A_\ell^T\, M_\ell = A_\ell^+$, a pseudoinverse of A_ℓ. We refer to this particular version as SEQ-PINV, and we note that if A_ℓ has full rank then this corresponds to the choice $M_\ell = (A_\ell A_\ell^T)^{-1}$ in the above algorithm. For arbitrary rank, the pseudoinverse A_ℓ^+ must satisfy the two conditions $A\, A_\ell^+ A = A$ and $(A_\ell^+ A)^T = A_\ell^+ A$. Given the pivoted Cholesky factorization [31]

$$A_\ell A_\ell^T = \Pi_\ell \begin{pmatrix} R_\ell & S_\ell \\ 0 & 0 \end{pmatrix}^T \begin{pmatrix} R_\ell & S_\ell \\ 0 & 0 \end{pmatrix} \Pi_\ell^T, \qquad R \in \mathbb{R}^{r_\ell \times r_\ell}, \tag{3.2}$$

where Π is a permutation matrix and $r_\ell = \operatorname{rank}(A_\ell)$, such a pseudoinverse is given by

$$A_\ell^+ = A_\ell^T\, \Pi_\ell \begin{pmatrix} (R_\ell^T R_\ell)^{-1} & 0 \\ 0 & 0 \end{pmatrix} \Pi_\ell^T. \tag{3.3}$$

[3]While "BLOCK-SEQUENTIAL" might be a more descriptive generic name for the algorithm, we use "BLOCK-IT" for historical reasons.

The use of this pseudoinverse is feasible, e.g., when the blocks are small or when most of the rows of A_ℓ are close to structurally orthogonal leading to a very sparse (often banded) $A_\ell A_\ell^T$. The latter case is discussed further in Section 3.3.

We emphasize that we use the same scaling matrix T for all blocks. An alternative is to define scaling matrices T_ℓ for each block, either using the definitions from above with A_ℓ instead of A, or using ideas from [10]. It is our experience, however, that all these variants have the same performance in practice, when used together with a near-optimal choice of the relaxation parameter. We therefore choose to avoid the extra computing cost and memory usage that is required for an algorithm where the blocks are processed with different scaling matrices.

When $p = 1$ (only one block) then the algorithm Block-It is identical to the particular SIRT method used on the blocks, while Seq-Pinv produces the undesired least squares solution in one iteration. On the other hand, with the choice $p = m$ (i.e., each block is a single row) then — for the case where we use Cimmino's method on the blocks — we obtain Kaczmarz's method.

3.2. Block-Parallel Methods. Common for these methods is that they simultaneously process all the blocks of A (similar to how SIRT treats the rows of A). Two block-parallel methods are natural: either use ART-sweep on each block, or use the pseudoinverse A_ℓ^+ of the block matrix as explained above. It is also possible to use SIRT on each block, but ART is preferable because of its faster convergence, cf. the right part of Fig. 2.2.

Consider the case where we apply an ART method to all p blocks in parallel using the *same* starting vector x^{k-1}, producing the p intermediate result $x^{k,\ell}$ for $\ell = 1, \ldots, p$. The next iteration vector x^k is then obtained by combining these intermediate vectors, and different choices yield different block-parallel methods. The algorithm[4] takes the following form [11]:

Algorithm: SAP (String-Averaging-Projections)
Initialization: choose an arbitrary $x^0 \in \mathbb{R}^n$
Iteration: for $k = 0, 1, 2, \ldots$
  for $\ell = 1, \ldots, p$ execute in parallel
    $x^{k,\ell} = \text{ART-sweep}\,(\lambda, A_\ell, b_\ell, x^{k-1})$
  $x^k = {}^1\!/_p \sum_{\ell=1}^p x^{k,\ell}$.

We obtain the corresponding Par-Pinv algorithm by replacing the innermost step with

$$x^{k,\ell} = P_{\mathcal{C}}\big(x^{k-1} + \lambda\, A_\ell^+ (b_\ell - A_\ell\, x^{k-1})\big)$$

where A_ℓ^+ is the pseudoinverse from (3.2)–(3.3). If $p = m$ (each block is a single row) and $P_{\mathcal{C}}$ is the identity then both algorithms become the Cimmino method. If $p = 1$ then the former algorithm becomes ART while the latter produces the least-squares solution in one step.

Gordon and Gordon [23] introduced an alternative way to combined the intermediate vectors $x^{k,1}, \ldots, x^{k,p}$, and referred to their version of the algorithm as CARP (Component-Averaged Row Projections). In our framework, this corresponds to replacing the last step in the above SAP algorithm with

$$x^k = \sum_{\ell=1}^p D^\ell\, x^{k,\ell}$$

[4]While "Block-Parallel" might be a more descriptive generic name for the algorithm, we use "SAP" for historical reasons.

with the diagonal matrices $D^\ell = \mathrm{diag}(\delta_j^\ell/\nu_j)_{j=1}^n$, where

$$\delta_j^\ell = \begin{cases} 1, & \text{if } A_\ell(:,j) \neq 0, \\ 0, & \text{else,} \end{cases} \qquad \nu_j = \sum_{\ell=1}^{p} \delta_j^\ell \,, \tag{3.4}$$

and $A_\ell(:,j)$ denotes the jth column of A_ℓ. CARP reduces to the DROP algorithm when $p = m$.

3.3. The Role of Structurally Orthogonal Rows. Recall that each row of A corresponds to a single ray. In the appendix, we discuss the issue of row ordering with a focus on a parallel-beam setup where a "projection" corresponds to data from a set of parallel rays of a specific orientation. In particular, we consider random, multilevel and a simple heuristic ordering of the projections.

In all remaining experiments of this paper we will use our heuristic scheme for computing the projection ordering. We do this to have a fair comparison of all methods and avoid the risk of an "unfortunate" random selection of rows.

The schemes described in the appendix for ordering the rows of A according to the access of projections leaves the order of the rows belonging to each projection unchanged. A subsequent partitioning of A into p equal-sized blocks $A_1, A_2, \ldots, A_p$ then results in blocks having rows in arbitrary order from one or more projections. In this section we discuss an alternative special partitioning of A, where the rows in each block are carefully selected to be structurally orthogonal.

Consider the case of a block A_ℓ that consists solely of *structurally orthogonal* rows, i.e., the nonzero elements of each row are located such that $a_i^{\mathrm{T}} a_j = 0$, for all $i \neq j$. As an example with $n = 10$:

a_i		×				×		×		
a_j				×			×			×

Since there is no overlap between locations of nonzeros in a_i and a_j, we have $a_i^T x^{k,i-1} = a_i^T x^{k,0} = a_i^T x^{k-1}$ in ART-SWEEP. Therefore, for such a block, the simultaneous row projections of SIRT are mathematically equivalent to performing the row projections in any sequential order, and hence the resulting x^k is identical to that of ART. In fact, $A_\ell A_\ell^T$ is diagonal and hence SIRT is also identical to the use of A_ℓ^+.

The updates in each step of ART-SWEEP involve different elements of the iteration vector (in the above example, step i and j involve elements $2, 6, 8$ and $4, 7, 10$, respectively), and therefore all updating steps can be performed in parallel. It follows that we can implement an embarrassingly parallel ART method for such an A, in which the required structural orthogonality is implied:

Algorithm: $x^k \leftarrow$ **PART-SWEEP** (λ, A, b, x^{k-1})

for $i = 1, \ldots, m$ execute in parallel

$$x^k = P_{\mathcal{C}}\left(x^{k-1} + \lambda \, \frac{b_i - a_i^T \, x^{k-1}}{\|a_i\|_2^2} \, a_i \right).$$

If all blocks A_ℓ in a partition of A consists of structurally orthogonal rows then we arrive at the PART algorithm [22], which is identical to BLOCK-IT in Section (3.1) if the innermost step is replaced by

$$x^{k,\ell} = \text{PART-SWEEP}\,(\lambda, A_\ell, b_\ell, x^{k-1}).$$

Similar ideas have also been used by Bramley and Sameh [5] for solving linear systems of equations obtained from partial differential equations. Popa [45] introduced an

algorithm for partitioning a sparse matrix so that each block contains mutually (not necessarily structurally) orthogonal rows.

As mentioned, an iteration of this algorithm results in a iterate that is identical to the iterate from ART-SWEEP. However, in contrast to ART, the algorithm also allows for a simple and efficient parallel implementation on modern platforms, as long as a reasonable partitioning that meets the orthogonality requirement is available. In our 3D tomographic reconstructions, the sparsity in a row is $O(n^{1/3})$ nonzeros out of n elements. This makes structural orthogonality between the rows of A very pronounced and allows us to obtain the required partitioning, which has sufficiently many rows per block to be able to fully utilize the computational resources.

In order to obtain the required block partition, we apply a geometric heuristic based on the experimental setup that sorts rays with overlapping traces into distinct groups. This approach corresponds to the ideas in [22] for 2D reconstruction, and we refer to this work for details.

We emphasize that the orthogonality of rows can lead to a significant reduction in the locality of data when computing x^k and therefore does not represent optimal cache usage on a modern CPU from a computational point of view.

4. Comparison of the Block Methods. In this section we present an experimental comparison of the block methods in terms of the observed convergence behavior and runtime measurements, for simulated 3D reconstruction problems. In all experiments, we aim for fairness by adopting the particular relaxation parameter λ that is optimal for each method, and which will depend on the particular geometry and the number of blocks. One iteration performed by a method corresponds to one pass through all the equations in (1.1). All the methods discussed here are implemented to handle general setups, where the projection images are assumed to be obtained from arbitrary directions. In all the examples $P_{\mathcal{C}}$ is the orthogonal projection on the positive orthant. Our implementations explicitly construct and use the sparse matrix A, which is stored in main memory (in contrast to matrix-free approaches). The sparse-storage format used for A is a variant of the sliced ELLPACK format [38] which in our cases achieves significantly better memory bandwidth than the CSR format.

In the performance study we will consider both single-core and multi-core performance. We implemented the multi-threaded execution using OpenMP. All experiments were conducted on one of two different platforms: Intel Core i7-3820 @ 3.60GHz with 4 cores, cache sizes 32KB/256KB/10MB, and Hyper-Threading turned on, and a 2 socket Intel Xeon CPU E5-2680 v2 @ 2.80GHz with a total of 20 cores, cache sizes 32KB/256KB/24MB, and Hyper-Threading swithed off. On the former platform we use double precision (64-bit floating-point) in all calculations, while on the latter we use single precision (32-bit floating-point).

4.1. Model Problems. In order to compare the convergence behavior of the block methods, we compute the reconstruction of a known phantom object and monitor the error histories. We use a 3D version of the Shepp-Logan phantom [46], which has been repeatedly used in the literature as a benchmark. The experimental setup is modeled by parallel-beam sources distributed in all directions according to the Lebedev quadrature points [43] shown in Fig. A.2. We experimented with circular and spherical setups as well, and with cone-beam rays, but such variations do not lead to significant changes in comparison to the results presented here.

From the phantom object and the experimental setup we are able to generate any number of simulated projections. Specifically, three model problems are considered:

- a *small problem* of 16^3 voxels using 13 projections of 16×16 pixels,
- a *medium problem* of 128^3 voxels using 115 projections of 128×128 pixels,
- a *large problem* of 256^3 voxels using 133 projections of 256×256 pixels.

When setting up the sparse linear system $A\,x \simeq b$, the line integrals generating A are determined after the discretization of the phantom so that the number of equations m is equal to the total number of pixels available in the projections, and the number of unknowns n is equal to the number of voxels. Consequently, the resulting matrices A have dimensions 3,328 × 4,096 for the small problem, 1,884,160 × 2,097,152 for the medium problem, and 8,716,288 × 16,777,216 for the large problem, so that all systems are quite underdetermined. Given the exact solution $\bar{x}$ (the phantom image), the exact right-hand side $\bar{b} = A\,\bar{x}$ is contaminated by additive white Gaussian noise e, i.e., $b = \bar{b} + e$, scaled such that the relative noise level $\|e\|^2/\|\bar{b}\|^2$ is fixed at 0.05.

4.2. Convergence Behavior. In our first series of experiments we compared the convergence behavior for different block partitionings for the six block methods described in Sections 3.1–3.3 using the small problem. We partitioned A into $p = 1, 2, 4, \ldots, 64$ equal-sized blocks and we used algorithm ORP to determined the optimal fixed relaxation parameter λ for a target relative error $\eta_{\text{target}} = 0.15$ for all six methods. Figure 4.1 shows the corresponding error histories for 50 iterations using the optimal λ values. The convergence behavior of the basic methods SIRT and ART can be identified as the Block-It and SAP histories, respectively, for $p = 1$. The curve for Block-It using $p = 13$ (i.e., one block per projection) corresponds to the SART method of Anderson and Kak [2].

We see that the Block-It method quickly approaches the ART convergence behavior as p is increased; it becomes indistinguishable from ART for $p = 8$. The reason for this fast improvement is that the rows within each block quickly become structurally orthogonal when the blocks are reduced in size (as mentioned earlier this turns SIRT into ART). For the same reason, SAP approaches the SIRT convergence behavior as p increases, but here the break-even point is at $p = 16$. The SAP and CARP algorithms have almost identical convergence behavior for p smaller than the number of projections. This is because all elements of x^k are updated when treating an individual block. For large p, however, many elements of x^k are left unchanged when treating a block, in which case the CARP combination of the intermediate results $x^{k,1}, \ldots, x^{k,p}$ prevails as the better choice.

Another observation is that the use of the pseudoinverse in the block methods significantly alters the convergence behavior. For less than 4 blocks, the Seq-Pinv method does not reach the semi-convergence solution but converges directly to the unwanted noisy solution. For more than 4 blocks, it converges as fast as ART and has the same optimal relaxation parameter as ART. Similarly, in the case of the Par-Pinv method we see that the target relative error is not reached in 50 iterations for less than 8 blocks; for 8 and more blocks the convergence is slow compared to the other results. We conclude that for the 3D reconstruction problems considered here, the pseudoinverse block methods are not competitive with the other block methods, at least from a purely performance and memory usage perspective.

Finally we see that the PART method exhibits identical convergence behavior for all block partitionings; this is also true in the next series of experiments and we postpone the explanation to the discussion of those results below.

To validate the observed convergence of the block methods for larger problems, we performed a similar series of experiments for the large test problem with a target relative error set to 5% above the global minimum, see Table 4.1 for details. We

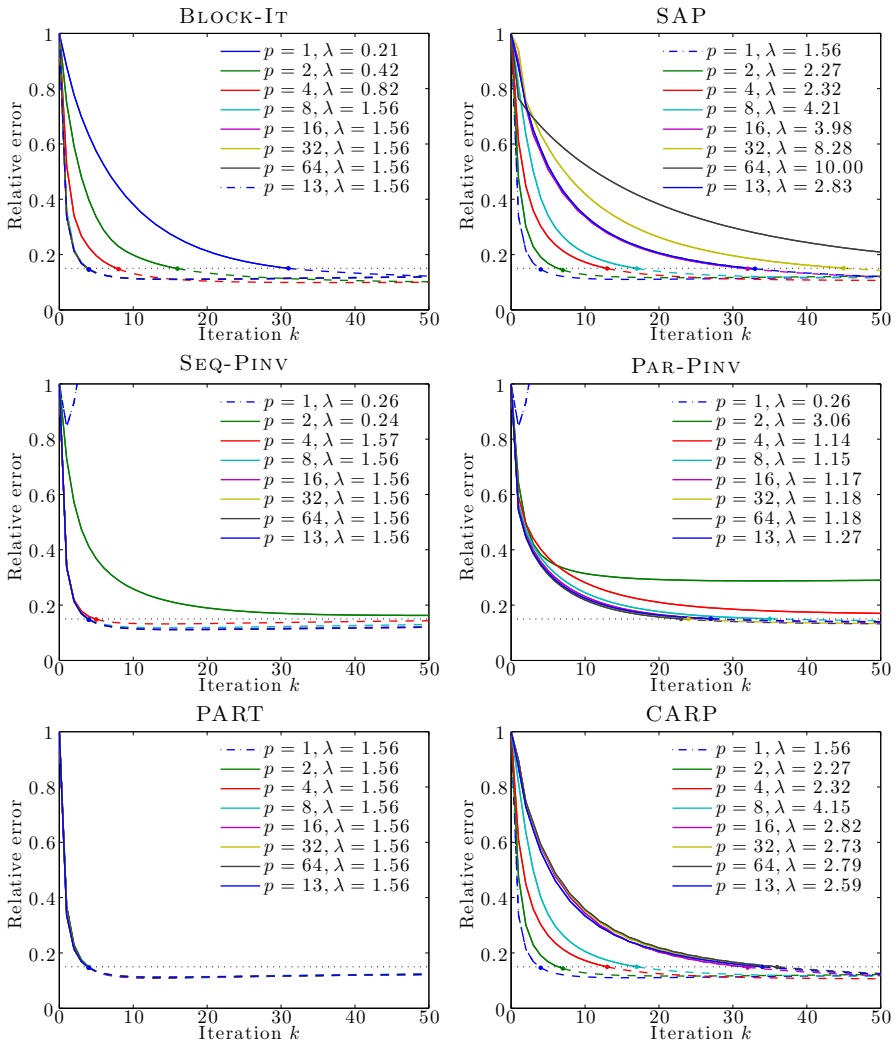


FIG. 4.1. *Error histories of the different block methods for the* ***small problem*** *calculated using 4 threads. The legends indicate the number of blocks p and the value of the corresponding optimal relaxation parameter λ. Note that some of the curves are indistinguishable from each other.*

TABLE 4.1

Results for the ***large problem****: the smallest relative error* $\eta_{\min}$ *in the first step of Alg. ORP from Sec. 2.3. We use these values to set the overall target error to* $\eta_{\text{target}} = 1.05 \cdot 0.196 = 0.206$*, i.e., 5% above the global minimum, when comparing the different methods. The combinations of methods and block sizes that do not reach this* η_{target} *in 50 iterations are not taken into account.*

Blocks p	1	2	4	8	16	32	64	133	256	532
Block-It	0.201	0.203	0.204	0.205	0.201	0.200	0.200	0.200	0.200	0.200
SAP	0.200	0.199	0.199	0.197	0.197	0.198	0.199	0.206	0.213	0.258
CARP	0.201	0.201	0.197	0.196	0.198	0.198	0.199	0.207	0.213	0.218
PART	0.200	0.200	0.199	0.200	0.200	0.200	0.200	0.199	0.199	0.199

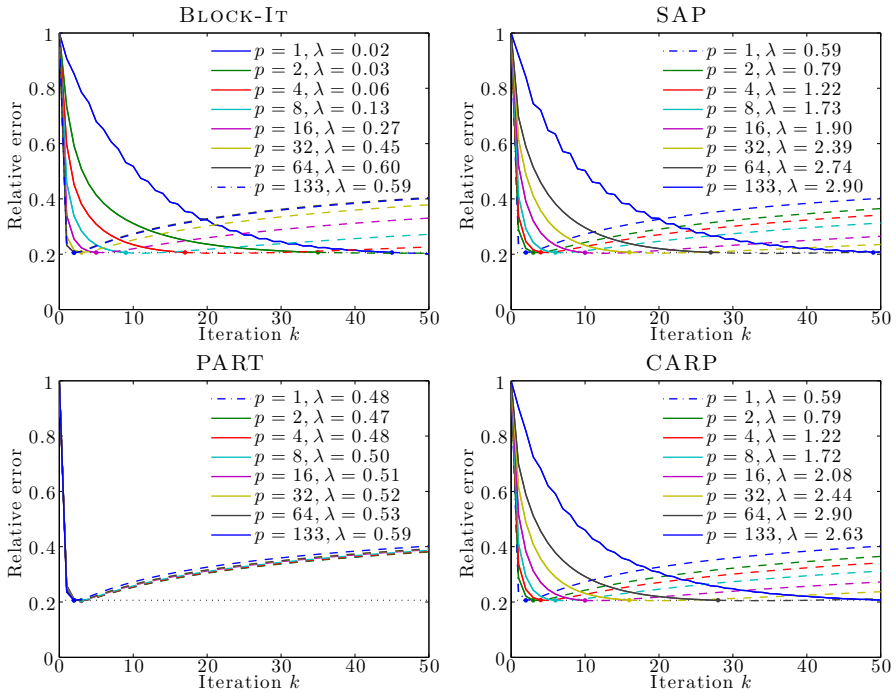


FIG. 4.2. *Error histories of the different block methods for the* ***large problem*** *calculated using 64 threads. The legends indicate the number of blocks p and the value of the corresponding optimal relaxation parameter λ. Note that some of the curves are indistinguishable from each other.*

consider only those block methods that do not involve a pseudoinverse. Figure 4.2 shows the error histories, from which we see that the general convergence trends are consistent with the previous observations. The main differences compared to the small problem are that:

1. Less iterations are required for ART to reach the target error.
2. More iterations are required for SIRT to reach the target error.
3. More blocks are required for Block-It before its convergence behavior is indistinguishable from that of ART.

We also note that, in general, lower values of the optimal fixed relaxation parameter are obtained.

PART exhibits almost identical convergence behavior for all block partitionings. This is quite remarkable since the rows in the blocks are not strictly structurally orthogonal (although almost so for $p = 133$ blocks). The reason for the identical convergence is that the matrix A is extremely sparse for the problem considered; there is on average only $\texttt{nnz}/m \approx 302$ nonzeros for each row of $n = 256^3 = 16{,}777{,}216$ elements. At the same time the computation is divided between 64 independent threads, which are assigned chunk sizes of 200 consecutive rows in a static manner. This significantly reduces the risk of threads working on the same columns at the same time. However, we stress that the method is not equivalent to ART (and neither deterministic nor robust) for other than $p = 532$ blocks which, for this example, is the number of blocks required to have structurally orthogonal rows in all blocks.

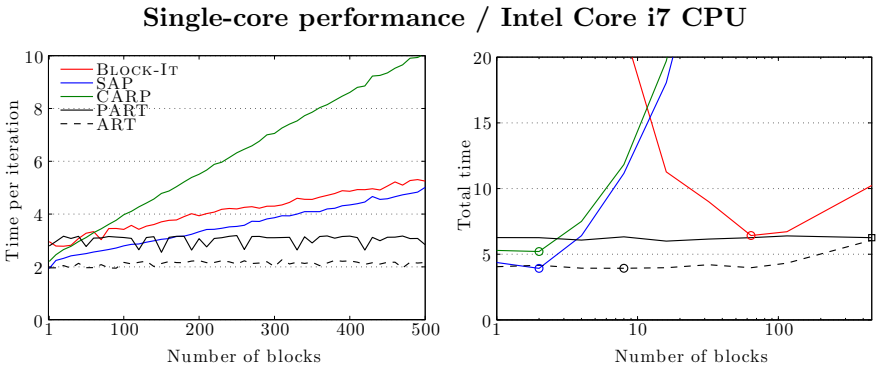


FIG. 4.3. *Computing time per iteration and total reconstruction time in seconds for the* ***medium problem*** *as a function of the number of blocks. All methods use the optimal* λ *and* η_{target} *set at* 5% *above the global minimum of all methods. The optimal number of blocks for each method is indicated by circles. In the case of* PART *the required minimum number of blocks* $p = 460$ *is indicated as a square. All timings are for a single thread.*

4.3. Single-Core Performance. Figure 4.3 shows timing results for single-core computations using the medium test problem. The left figure shows the time per iteration as a function of the number of blocks for the different block methods.

For reference we also show the results for a block sequential implementation of ART, from here on denoted by "ART (1 thread)," where the single-block case is the implementation commonly associated with ART. The reason for this is that the underlying sliced ELLPACK sparse storage of the matrix A introduces zero-padding in order to eliminate row pointers, which in turn increases the data throughput but reduces the cache efficiency due to padded bytes. This means that even the sequential ART method, rather than working on only one block, might improve from using a block partitioning (slices in the storage format), because the zero-padding per block can be reduced this way with little overhead in the algorithm.

We see from the left figure that there is considerable overhead associated with partitioning into blocks when p is large, in particular for the CARP algorithm. The SAP algorithm performs much better for increasing number of blocks. We also observe that Block-It has longer time per iteration than SAP for the same number of blocks.

The main influence on these results is the cache utilization achieved in the different algorithms. The algorithms that apply ART in each block have significantly better cache utilization than than those based on SIRT because of the data locality of the row a_i, which is accessed twice in each iteration, and its consecutive row a_{i+1}, which is always accessed next. For SIRT every row is accessed only once in the multiplication by A, followed by a separate access of every row in the multiplication by A^T, at which time the cache lines from the first access are most likely flushed. In practice, this produces the difference between the slopes of the curves for Block-It and PART.

The right part of Fig. 4.3 shows the total reconstruction time as a function of the number of blocks, for which the minimum of each curve is marked by a circle (and $p = 460$ for PART) in order to indicate the fastest possible reconstructions for each method. We see that the methods typically have a specific number of blocks that provide the optimal partitioning in terms of reconstruction time for a particular problem. Table 4.2 on p. 16 gives a quick visual comparison of the reconstructed images obtained by the different methods using a single core, along with information about the performance of the algorithms. All the reconstructions are computed for a

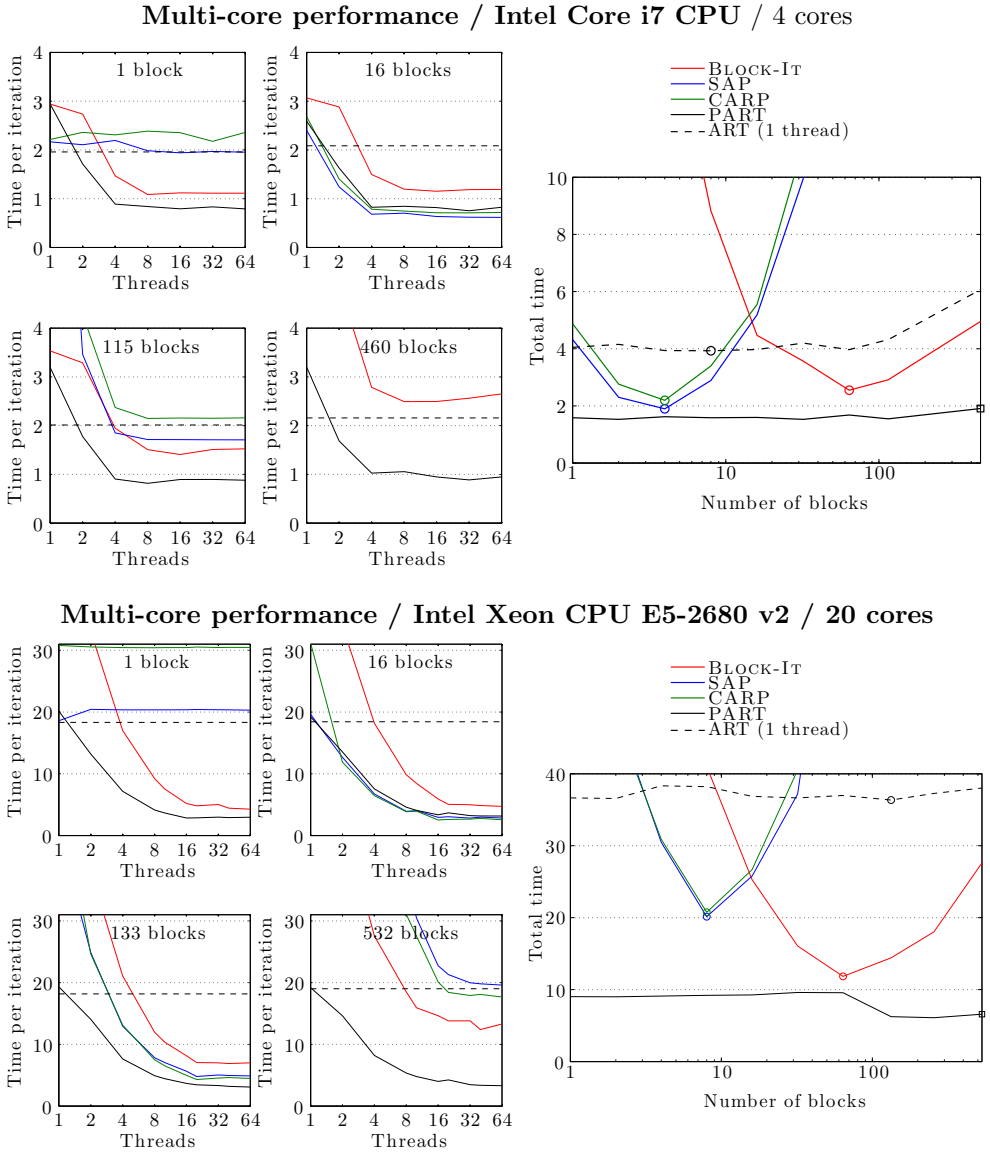


FIG. 4.4. *Right parts are the same as in Fig. 4.3 but for multi-core platforms. Top left: the time per iteration as a function of threads for the* ***medium problem*** *using partitionings of p =1, 16, 115, and 460 blocks. Bottom left: similar to the top left but for the* ***large problem*** *using partitionings of p =1, 16, 133, and 532 blocks. We use 4 and 20 cores; all timings of the total time are for 64 threads except ART which uses one thread.*

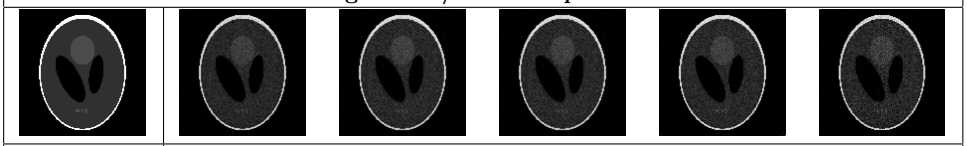
target relative error set to 5% above the global minimum and no distinct difference is observed. On the basis of the presented results, we conclude that the fastest methods in a single-core setting — due to superior cache usage — are ART and SAP.

4.4. Multi-Core Performance. We now study the performance of the block methods by considering the medium and large test problems on multi-core platforms. The top left half of Fig. 4.4 shows the time per iteration as a function of the number of threads for the medium problem, for partitionings into 1, 16, 115, and 460 blocks,

TABLE 4.2

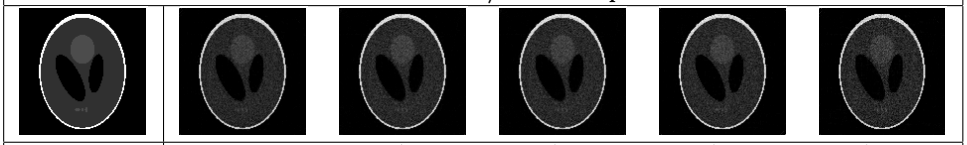
The central slice of the 3D Shepp-Logan phantom (leftmost images) and the corresponding reconstructions produced by the different block methods and ART. The top and middle parts show results for the ***medium problem*** *using 1 and 4 cores, respectively, on the Intel Core i7 CPU. The bottom part shows results for the* ***large problem*** *computed on the Intel Xeon CPU E5-2680 v2 using 20 cores. We also list the number of blocks, the number of iterations, and the total reconstruction time for the fastest reconstruction of each method using the optimal fixed relaxation parameter.*

Single-core / medium problem



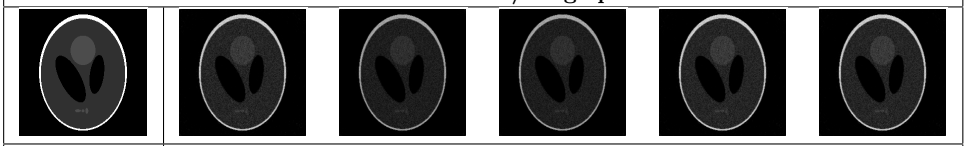
Method	BLOCK-IT	SAP	CARP	PART	ART
Blocks	64	2	2	460	8
Iterations	2	2	2	2	2
Time (s)	6.41	**3.92**	5.20	6.25	**3.92**

Multi-core: 4 cores / medium problem



Method	BLOCK-IT	SAP	CARP	PART	ART
Blocks	64	4	4	460	8
Iterations	2	3	3	2	2
Time (s)	2.54	**1.89**	2.19	**1.90**	3.92

Multi-core: 20 cores / large problem



Method	BLOCK-IT	SAP	CARP	PART	ART
Blocks	64	8	8	532	133
Iterations	2	6	6	2	2
Time (s)	11.84	20.14	20.75	**6.58**	34.54

running on an Intel Core i7 CPU with 4 cores. In most cases the time per iteration is reduced by a factor of about 3 by using four or more threads compared to using only one in the multi-threaded implementations. The reason that we do not see better scaling (i.e., a factor of 4) is a combination of mainly three issues; the overhead from the block partitioning (cf. Fig. 4.3), the overhead from the forking, joining and scheduling of the threads with OpenMP, and the less optimal cache usage (all threads share the same L3 cache while working on different parts of the A matrix).

The top right half of Fig. 4.4 shows the total reconstruction times for the medium problem as a function of the number of blocks. Again we observe an optimal number of blocks for each of the different block methods.

With the exception of ART and PART, the performance of each method is very sensitive to the choice of the number of blocks. As we move away from the optimal number of blocks, the timings deteriorate rapidly. This means that on different problems and/or different number of cores, the number of blocks would have to be fine-tuned for each case in order to obtain the best results. On the other hand, ART and PART are very stable in this respect and no such tuning is required.

The key results are summarized in Table 4.2 along with a visualization of the reconstructed objects; there is no significant difference in the obtained images. We see that in a multi-core setting with 4 cores the fastest reconstruction time is achieved by SAP in the case where it uses $p = 4$ blocks and three iterations to reach the target relative error. As a close runner-up we have the PART using the required $p = 460$ blocks and only two iterations. Tests for other problem sizes on the 4-core platform show that the observed ranking of the methods is consistent.

In our final series of experiments, we repeat the computations on a 2 socket Intel Xeon CPU E5-2680 v2 platform with 20 cores. The results are shown in the bottom half of Fig. 4.4 and also summarized in the bottom part of Table 4.2. The left figures show that using 64 threads reduces the time per iteration by a factor of 5–10 for the multi-threaded implementations of the block methods compared to using 1 thread. The reduction depends on the particular method and the number of blocks used in the partition. In particular the PART algorithm scales well for more threads even for partitionings with many blocks.

From the reconstruction times presented in the right half of Fig. 4.4 we observe a different performance trend for the block methods on a multi-core machine with 20 cores instead of 4 cores. We see that the block-parallel methods perform notably worse in comparison to the other methods on a 20-core platform. We explain this by the unavoidable trade-off in these methods between having fast time per iteration, which requires up to 20 blocks for parallel execution, and having fast convergence, which requires a partition of much fewer blocks (cf. Fig. 4.2).

The bock-sequential Block-It method performs better than the block-parallel methods on 20 cores but suffers from bad cache utilization as already pointed out. The PART method has the fastest reconstruction time with a distinct improvement around 100 blocks, which indicates a shift from 3 to 2 necessary iterations as the rows in each block gets closer to satisfying structural orthogonality. We conclude that for our test problems the PART algorithm is best suited for the level of multi-threading required by a high number of computing cores.

Comparing the execution times to other works is difficult since most authors use a circular setup for sliced 3D reconstructions (e.g., fixed tilt-angle geometries). Sliced approaches are significantly cheaper in terms of computational work and memory usage due to symmetries in the setup. However, a recent GPU-based study [25] of 3D reconstruction from arbitrarily aligned projections with artifact reduction reports a time per iteration of 4.0 s for a problem of 128^3 voxels using 50 projections of 256×256 pixels in single precision. Comparing with our medium problem (128^3 voxels using 115 projections of 128×128 pixels in double precision) we see that our multi-core CPU implementation of the PART algorithm achieves times per iteration very similar to the GPU implementation presented in [25].

5. Conclusion. We describe the central aspects of implementing block algebraic iterative reconstruction methods for tomographic imaging problems on multi-core platforms. Our implementations suit arbitrary imaging geometries in which the projection images can be obtained from any direction and the matrix describing the geometry is explicitly constructed or read from disk and stored in memory. We present a framework for block-sequential and block-parallel methods, and the most promising algorithms are carefully implemented for good performance. In order to make a fair comparison between the different methods we use a training procedure to obtain the optimal fixed relaxation parameter for a particular method, using a model problem. We compare the performance of the block methods to ART in a number of numerical

TABLE 5.1
Summary of the most efficient implementations on different computer platforms for the underdetermined model problems considered in this work.

Platform	**Most efficient implementations**
Single-core CPU	ART and SAP
Multi-core CPU: 4 cores	SAP and PART
Multi-core CPU: 20 cores	PART (or BLOCK-IT)

experiments in both single-core and multi-core settings.

Our work focuses on 3D parallel-beam tomographic reconstruction; however, other results for cone-beam setups (not reported here) confirm our observations. Our numerical results support the common observation that ART has the fastest convergence during the semi-convergence phase. In addition, we show that the fast convergence can also be achieved in both block-sequential and block-parallel methods if the number of blocks is chosen properly. At the same time the level of parallelism introduced in the block methods is sufficient for multi-core platforms. We also demonstrate that the cache utilization of an actual implementation of ART is significantly better than that of an implementation of SIRT. We conclude that ART is the best choice for single-core execution and the best choice to use in a block method on the individual blocks.

Table 5.1 summarizes our main results for single- and multi-core platforms. For single-core and few-core execution the block-parallel methods are the fastest. As the number of cores increases, the block-sequential methods perform better and, in particular, the PART algorithm — which can utilize structurally orthogonal rows — is well suited for current high-end and future multi-core platforms. If structural orthogonality cannot be utilized then BLOCK-IT is preferable.

We note that the main results in Table 5.1 are obtained for model problems that produce underdetermined systems; for other problems the most efficient approach can be different from the ones we found. For example, as illustrated in a recent study by Karonis et al. [34], the CARP method has shown great applicability in large-scale settings if the number of equations is significantly larger (up to two orders of magnitude) than the number of voxels, because then the overhead from combining the current solutions of all threads after each iteration is significantly less costly.

Acknowledgements. We thank Tommy Elving and the two referees for many valuable comments and suggestions that helped to improve the presentation.

Appendix A. Row Orderings.

The particular form of the matrix A depends on the geometry of the underlying tomography problem. The ordering of its rows corresponds to the order in which the rays are organized. We say that the *natural ordering* of the rows arises from treating rays according the position of the pixels in the projections, one projection after the other. Several investigations [26], [30], [39] demonstrate that the row ordering has a strong effect on the practical performance of algebraic reconstruction techniques, in particular when the angles that define the projections cover an (almost) full range. In that respect, the natural ordering is rarely optimal for the algebraic reconstruction methods that are the focus in this work.

The convergence of ART is fast when the rows of A are close to orthogonal [26], as reflected in the fact that the smaller the condition number of A, the faster the convergence, cf. (2.1). Figure A.1 gives a geometric illustration of this point for

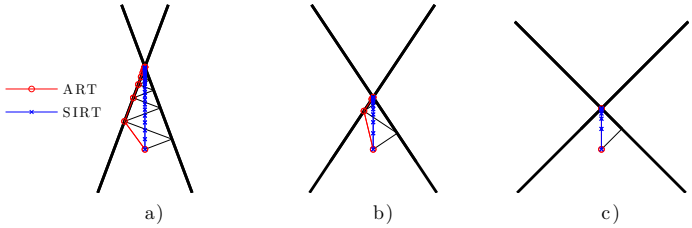


FIG. A.1. *Convergence of ART and SIRT for an example with $m = n = 2$, where the corresponding hyperplanes/lines are a) far from orthogonal, b) close to orthogonal, and c) orthogonal.*

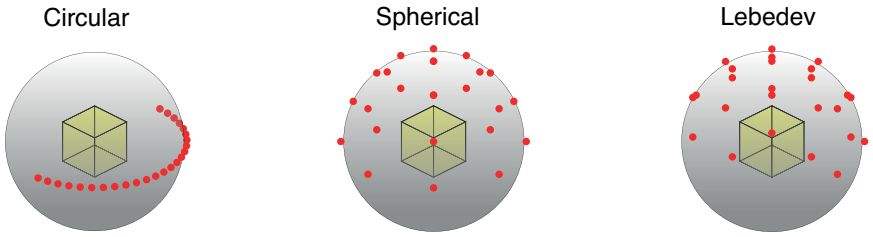


FIG. A.2. *Three different setups for parallel-beam 3D tomography, showing 25 points on the unit sphere that define the directions of the parallel rays. Left: the sources are equidistantly distributed on a half-circle (suited only for computing slices of a 3D reconstruction). Middle: the sources are distributed equidistantly in latitude and longitude on the top half of the sphere. Right: the sources are distributed according to Lebedev quadrature points that give an approximately uniform distribution on the sphere.*

$m = n = 2$ (i.e., two equations), where the solution is the intersection of two lines in the plane. We show three cases, where the lines are far from orthogonal, close to orthogonal, and orthogonal, respectively. In the first case each iteration approaches the intersection slowly since the directions of the steps are close to parallel. The third case is clearly optimal since the second step is orthogonal to the first, and the solution is found in one iteration.

In general, each of the m linear equations represents a hyperplane and for a consistent system the solution x^* is a point on the intersection of the hyperplanes. The convergence of ART is fast if consecutive rows represent hyperplanes that are almost orthogonal to the space spanned by the previous rows. Therefore, for ART we could reorder the rows such that for all $i = 1, \ldots, m-1$ row a_i is chosen to be the row among $a_{i+1}, \ldots, a_m$ that has the largest angle to $\text{span}\{a_1, \ldots, a_{i-1}\}$. This is possible as long as $\text{rank}(A) \geq m$. However, this "greedy algorithm" leads to a combinatorial problem that is costly and not feasible to perform in practice except for very small matrices.

A more practical approach is to adopt the natural ordering for the rays belonging to the individual projections and only consider the order in which the projections are accessed. As reported elsewhere (see [30] and references therein), and confirmed by our testing, it has little effect to reorder rays within projections. The goal is therefore to arrange the projections in an order where the information in the next projection is as independent of the previously accessed projections as possible. The most common access order schemes proposed so far for 2D reconstructions are the multi-level scheme [26], the prime number decomposition [30], the Golden Section scheme [35], and the weighted distance scheme [39].

Figure A.2 shows three setups for 3D parallel-beam tomographic reconstruction. The directions of the parallel rays are indicated by P points on the unit sphere that represent the unit vectors $\hat{p}_i$, $i = 1, \ldots, P$. The projection images, one for each direction, are recorded on flat square detectors oriented perpendicular to the ray. The circular and spherical distributions are enumerable in a systematic manner in terms of equidistant angles while the Lebedev distribution,[5] by construction, is not.

In this work, we employ a simple heuristic to determine the ordering of the projections, which in practice is as effective as the schemes mentioned above. Our heuristic scheme has the advantage that it does not require a systematic positioning of the sources or an initial enumeration of the projections. First we set $\mathcal{I} = \{1, 2, \ldots, P\}$ and evaluate the sum $\bar{p}_{\mathcal{I}} = \sum_{i \in \mathcal{I}} \hat{p}_i$, and then we select the particular projection whose direction vector has the largest angular distance to $\bar{p}_{\mathcal{I}}$, i.e., the selected projection number j minimizes $\bar{p}_{\mathcal{I}}^T \hat{p}_j$ for $j \in \mathcal{I}$. At this stage the set of selected projections is $\mathcal{V} = \{j\}$.

Subsequent projection numbers are then chosen by maintaining and updating the set of selected projections $\mathcal{V}$ with $V = |\mathcal{V}|$ elements, together with the set of candidate projections $\mathcal{U} = \mathcal{I} \setminus \mathcal{V}$ with $U = |\mathcal{U}|$ elements. Each step has the following heuristic form:

1. Calculate the angular distances $c_{u,v} = |\hat{p}_u^T \hat{p}_v|$ between every candidate projection $\hat{p}_u, u \in \mathcal{U}$ and all previously selected projections $\hat{p}_v, v \in \mathcal{V}$.
2. Sort the angular distances belonging to the individual candidates in descending order so that $c_{u,v} \geq c_{u,v+1}$ for $u = 1, \ldots, U$ and $v = 1, \ldots, V - 1$.
3. For $k = 1, 2, \ldots, V$
   (a) Update the set of candidates to those projections that have the largest angular distance to the closest previously selected projection, i.e., $\mathcal{U} = \{1 \leq u \leq U \,|\, c_{u,k} = \min_\nu c_{\nu,k}\}$.
   (b) If $|\mathcal{U}| = 1$ then break.
4. If $|\mathcal{U}| = 1$ then select the projection as the single remaining candidate in $\mathcal{U}$. Otherwise, select a projection from $\mathcal{U}$ in the same manner as the initial projection, i.e., the one that minimizes $\hat{p}_j^T \sum_{i \in \mathcal{U}} \hat{p}_i$ for $j \in \mathcal{U}$. Purge the selected row number from $\mathcal{V}$.

This procedure is repeated until all rows have been treated, i.e., $V = 0$. The algorithm is specifically designed to handle the case $m > n$, but it also works for $m \leq n$.

Figure A.3 illustrates the effect of different projection orderings when using the ART method with an optimal λ for 3D reconstructions. The curves show the error histories for the natural ordering, the average of 100 random orderings, the multi-level scheme of Guan and Gordon [26], and our heuristic scheme. The three graphs show results corresponding to the three different ray distributions in Fig. A.2. In the circular and spherical setups, which exhibit systematic positioning of the sources, we see that the natural ordering represents the worst case; this is because the rows associated with the natural ordering of the projections resemble each other structurally. We also see that only a small increase in convergence rate is achieved, compared to the random ordering, by using the other ordering schemes. Using the Lebedev distribution of the rays, we reach the smallest error and the curves for random ordering and any ordering scheme become indistinguishable.

We conclude that none of the proposed schemes for ordering of the projections have significant advantage over a random ordering, and their performance becomes

[5] The Lebedev distribution [37] does not reflect a specific measurement geometry (to our knowledge); it is an artificial distribution that represent rays which are well distributed over all angles.

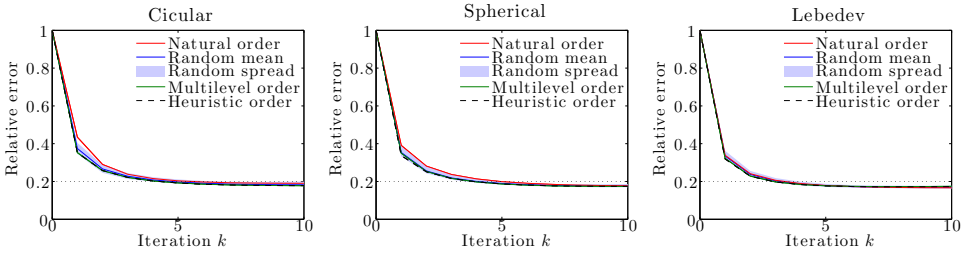


FIG. A.3. *Typical effect of the projection ordering for the ART method. The blue line shows the mean of 100 random orderings and the shaded area indicates the spread. The red line is the result without reordering (natural order). From left to right, three cases are shown corresponding to circular, spherical, and Lebedev distribution of rays, cf. Fig. A.2. The reconstructions have* 32^3 *voxels and* $P = 25$ *projections of size* 32^2 *pixels. In all cases we use the optimal relaxation parameter.*

practically identical as the number of projections increases. We have illustrated this for parallel rays, but our experiments show a similar behavior for the case of cone-beams where the rays associated with each projection originate from a single point.

REFERENCES

[1] R. AHARONI AND Y. CENSOR, *Block-iterative projection methods for parallel computation of solutions to convex feasibility problems*, Lin. Alg. Appl., 120 (1989), pp. 165–175.

[2] A.H. ANDERSEN AND A.C. KAK, *Simultaneous Algebraic Reconstruction Technique (SART): A superior implementation of the ART algorithm*, Ultrasonic Imaging, 6 (1984), pp. 81–94.

[3] M. BERTERO AND P. BOCCACCI, *Introduction to Inverse Problems in Imaging*, Institute of Physics Publishing, Bristol, 1998.

[4] M. BERTERO, D. BINDI, P. BOCCACCI, M. CATTANEO, C. EVA, AND V. LANZA, *Application of the projected Landweber method to the estimation of the source time function in seismology*, Inverse Problems, 13 (1997), pp. 465–486.

[5] R. BRAMLEY AND A. SAMEH, *Domain decomposition for parallel row projection algorithms*, Applied Numerical Mathematics, 8 (1991), pp. 303–315.

[6] G. CIMMINO, *Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari*, La Ric. Sci., XVI, Ser. II, Anno IX, 1 (1938), pp. 326–333.

[7] Y. CENSOR, *Parallel application of block-iterative methods in medical imaging and radiation therapy*, Mathematical Programming, 42 (1988), pp. 307–325. 10.1007/BF01589408.

[8] Y. CENSOR, D. GORDON, AND R. GORDON, *BICAV: An inherently parallel algorithm or sparse systems with pixel-related weighting*, IEEE Trans. Medical Imaging, 20 (2001), pp. 1050–1060.

[9] Y. CENSOR, D. GORDON, AND R. GORDON, *Component averaging: An efficient iterative parallel algorithm for large and sparse unstructured problems*, Parallel Computing, 27 (2001), pp. 777–808.

[10] Y. CENSOR AND T. ELFVING, *Block-iterative algorithms with diagonally scaled oblique projections for the linear feasibility problem*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 40–58.

[11] Y. CENSOR, T. ELFVING, AND G. T. HERMAN, *Averaging strings of sequential iterations for convex feasibility problems;* in D. Butnariu, Y. Censor, and S. Reich (Eds), *Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications*, Elsevier, Amsterdam, The Netherlands, pp. 101–114, 2001.

[12] Y. CENSOR, T. ELFVING, G. T. HERMAN, AND T. NIKAZAD, *On diagonally relaxed orthogonal projection methods*, SIAM J. Sci. Comput., 30 (2008), pp. 473–504.

[13] P. P. B. EGGERMONT, G. T. HERMAN, AND A. LENT, *Iterative algorithms for large partitioned linear systems, with applications to image reconstruction*, Lin. Alg. Appl., 40 (1981), pp. 37–67.

[14] T. ELFVING, *Block-iterative methods for consistent and inconsistent linear equations*, Numer. Math., 35 (1980), pp. 1–12. 10.1007/BF01396365.

[15] T. ELFVING, P. C. HANSEN, AND T. NIKAZAD, *Semi-convergence and relaxation parameters for projected SIRT algorithms*, SIAM J. Sci. Comput., 34 (2012), pp. A2000–A2017, DOI: 10.1137/110834640.

[16] T. Elfving, P. C. Hansen, and T. Nikazad, *Semi-convergence properties of Kaczmarz's method*, Inverse Problems, 30 (2014), DOI: 10.1088/0266-5611/30/5/055007.
[17] T. Elfving and T. Nikazad, *Properties of a class of block-iterative methods*, Inverse Problems, 25 (2009), 115011, DOI: 10.1088/0266-5611/25/11/115011.
[18] T. Elfving, T. Nikazad, and P. C. Hansen, *Semi-convergence and relaxation parameters for a class of SIRT algorithms*, Electronic Trans. on Numer. Anal., 37 (2010), pp. 321–336.
[19] T. Elfving, T. Nikazad, and C. Popa, *A class of iterative methods: semi-convergence, stopping rules, inconsistency, and constraining*, in: Y. Censor, M. Jiang, and G. Wang (Eds.), *Biomedical Mathematics: Promising Directions in Imaging, Therapy Planning, and Inverse Problems*, Medical Physics Publishing, Madison, WI, 2010.
[20] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems*, Kluwer, Dordrecht, The Netherlands, 1996.
[21] P. Gilbert, *Iterative methods for the three-dimensional reconstruction of an object from projections*, J. Theoretical Biology, 36 (1972), pp. 105–117.
[22] D. Gordon, *Parallel ART for image reconstruction in CT using processor arrays*, Int. J. Parallel, Emergent and Distrib. Syst., 21 (2006), pp. 365–380.
[23] D. Gordon and R. Gordon, *Component-averaged row projections: A robust, block-parallel scheme for sparse linear systems*, SIAM J. Sci. Comput., 27 (2005), pp. 1092–1117.
[24] R. Gordon, R. Bender, G. T. Herman, *Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography*, J. Theoretical Biology, 29 (1970), pp. 477–481.
[25] D. Gross, U. Heil, R. Schulze, E. Schoemer, and U. Schwanecke, *GPU-based volume reconstruction from very few arbitrarily aligned x-ray images*, SIAM J. Sci. Comput., 31 (2009), pp. 4204–4221.
[26] H. Guan and R. Gordon, *A projection access order for speedy convergence of ART (algebraic reconstruction technique): a multilevel scheme for computed tomography*, Physics in Medicine and Biology, 39 (2001), pp. 2005–2022.
[27] P. C. Hansen, *Discrete Inverse Problems: Insight and Algorithms*, SIAM, Philadelphia, 2010.
[28] P. C. Hansen and M. Saxild-Hansen, *AIR Tools – A MATLAB package of algebraic iterative reconstruction methods*, J. Comp. Appl. Math., 236 (2011), pp. 2167–2178, DOI: 10.1007/s10543-011-0359-8.
[29] G. T. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*, 2. Ed., Springer, New York, USA, 2009.
[30] G. T. Herman and L. B. Meyer, *Algebraic reconstruction techniques can be made computationally efficient*, IEEE Trans. Medical Imaging, 12 (1993), pp. 600–609.
[31] N. J. Higham, *Analysis of the Cholesky decomposition of a semi-definite matrix*, in: Reliable Numerical Computation, eds. M.G. Cox and S. Hammarling (Clarendon Press, Oxford, 1990) pp. 161–185.
[32] M. Jiang and G. Wang, *Convergence studies on iterative algorithms for image reconstruction*, IEEE Trans. Medical Imaging, 22 (2003), pp. 569–579.
[33] S. Kaczmarz, *Angenäherte Auflösung von Systemen linearer Gleichungen*, Bull. Acad. Pol. Sci. Lett., A35 (1937), pp. 355–357.
[34] N. T. Karonis, K. L. Duffin, C. E. Ordoñez, B. Erdelyi, T. D. Uram, E. C. Olson, G. Coutrakon, M. E. Papka, *Distributed and hardware accelerated computing for clinical medical imaging using proton computed tomography (pCT)*, Journal of Parallel and Distributed Computing, 73 (2013), pp. 1605–1612.
[35] T. Köhler, *A projection access schme for iterative reconstruction based on the golden section*, IEEE Nuclear Science Symposium Conference Record, 6 (2004), pp. 3961–3965.
[36] L. Landweber, *An iteration formula for Fredholm integral of the first kind*, Amer. J. Math. 73 (1951), p. 615–624.
[37] V. I. Lebedev, *Quadratures on a sphere*, USSR Computational Mathematics and Mathematical Physics, 16 (1976), pp. 10–24.
[38] A. Monakov, A. Lokhmotov, and A. Avestisyan, *Automatically tuning sparse matrix-vector multiplication for GPU Architectures*; in Y. N. Patt et al. (Eds.), *High Performance Embedded Architectures and Compilers*, Lecture Notes in Computer Science, 5952 (2010), pp. 111–125.
[39] K. Mueller, R. Yagel, and J. F. Cornhill, *The weighted-distance scheme: a globally optimizing projection ordering method for ART*, IEEE Trans. Medical Imaging, 16 (1997), pp. 223–230.
[40] F. Natterer, *The Mathematics of Computerized Tomography*, SIAM, Philadelphia, 2001.
[41] D. Needell and J. A. Tropp, *Paved with good intentions: Analysis of a randomized block Kaczmarc method*, Lin. Alg. Appl., 441 (2014), pp. 199–221. DOI: 10.1016/

j.laa.2012.12.022.

[42] Yu. Nesterov, *Introductory Lectures on Convex Optimization*, Kluwer Academic, Dordrecht, The Netherlands, 2004.

[43] R. Parrish, *getLebedevSphere*, Matlab file. `www.mathworks.com/matlabcentral/fileexchange/27097-getlebedevsphere` (2010).

[44] S. N. Penfold, R. W. Schulte, Y. Censor, V. Bashkirov, S. McAllister, K. E. Schubert, and A. B. Rosenfeld, *Block-iterative and string-averaging projection algorithms in photon computed tomography image reconstruction*, in: Y. Censor, M. Jiang, and G. Wang (Eds.), *Biomedical Mathematics: Promising Directions in Imaging, Therapy Planning, and Inverse Problems*, Medical Physics Publishing, Madison, WI, 2010.

[45] C. Popa, *Block-projections algorithms with blocks containing mutually orthogonal rows and columns*, BIT Numerical Mathematics, 9 (1999), pp. 323–338.

[46] M. Schabel, *3D Shepp-Logan phantom*, Matlab file. `www.mathworks.com/matlabcentral/fileexchange/9416-3d-shepp-logan-phantom` (2006).

[47] T. Strohmer and R. Vershynin, *A randomized Kaczmarz algorithm for linear systems with exponential convergence*, J. Fourier Anal. Appl., 15 (2009) 262–278.

[48] M. C. A. Van Dijke, H. A. van der Vorst, and M. A. Viergever, *On the relation between ART, block-ART and SIRT;* in A. E. Todd-Pokropek and M. A. Viergever (Eds.), *Medical Images: Formation, Handling and Evaluation*, NATO ASI Series F, Computer and Systems Sciences, 98 (1992), pp. 377–396.

[49] A. van der Sluis, H. A. van der Vorst, *SIRT- and CG-type methods for the iterative solution of sparse linear least-squares problems*, Lin. Alg. Appl., 130 (1990), pp. 257–302.

[50] X. Wan, F. Zhang, Q. Chu, K. Zhang, F. Sun, B. Yuan, and Z. Liu, *Three-dimensional reconstruction using an adaptive simultaneous algebraic reconstruction technique in electron tomography*, J. Structural Biology, 174 (2011), pp. 277–287.